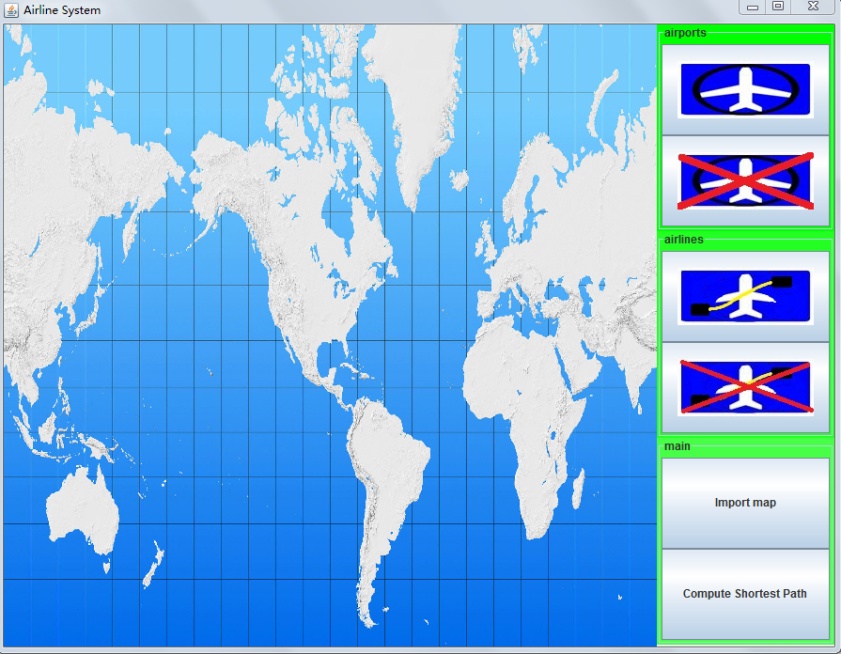
**Airline Routing System开发文档**

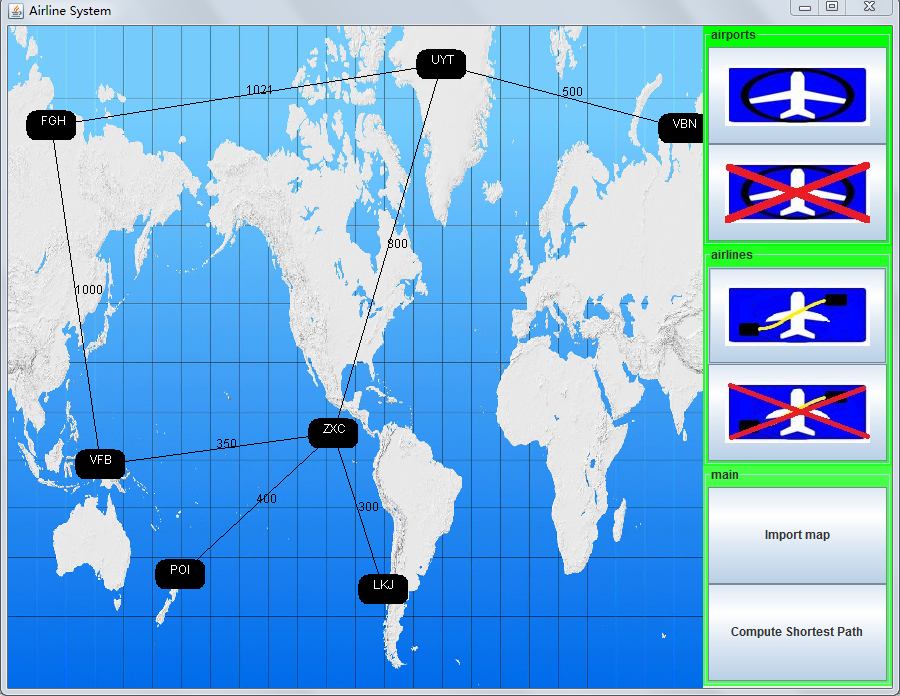
1. **基本功能**

**1.1导入图**

用户打开程序后会出现如下界面，用户可以点击main框中的import map按钮（即第5个按钮，按照提示导入文件，原始的测试文件名为airline.txt，



导入后出现如下界面

****

并且可以不用关闭程序，重新导入其他文件，文件格式为“VERTEX XXX”和“EDGE XXX YYY ZZZ”其中XXX、YYY为端点，ZZZ为航线长度。必须为数字。

支持加入、删除机场，加入、删除航线的功能，改变后的图形信息会被记录到原来导入的文件中。如果没有导入文件，直接加入机场、航线的，图形会被记录到airlines.txt中。

**1.2加入机场**

按第一个按钮可以实现加入机场功能，出现如下对话框，按照要求填入新的机场的名字，按确定即可。

如果用户输入的名字已经有了，就会警告“already have this vertex”

随后改变信息会被记录到文件中。

****

**1.3删除机场**

按第二个按钮可以实现删除机场功能，按照要求填入图上已有的机场的名字，按确定即可。

若输入的名称图上没有，则会弹出警告：“No Such Vertex”

随后改变信息会被记录到文件中。

**1.4加入航线**

按第三个按钮可以实现加入航线功能，按照要求格式“XXX YYY ZZZ”XXX、YYY为端点机场，ZZZ为航线长度，填入图上没有的航线的名字，按确定即可。

如果格式不对,会警告“wrong input”

随后改变信息会被记录到文件中。

**1.5删除航线**

按第四个按钮可以实现删除航线功能，按照要求格式“XXX YYY ZZZ”XXX、YYY为端点机场，ZZZ为航线长度，填入图上已有的航线的名字，按确定即可。

如果没有这条边，就会警告“No such edge”

随后改变信息会被记录到文件中。

**1.6计算最短路径**

按第六个按钮可以实现计算最短路径功能，按照要求格式“XXX YYY”XXX、YYY为端点机场，填入图上已有的两个要计算的机场的名字，按确定即可。

计算完成后会在图上用红色标出路线。

如果起点与终点间不连通，则会弹出信息“graph isn’t connected”。

改变了图上信息后，需要再次点击输入，才能计算出最新的路径。

1. **方法与实现**

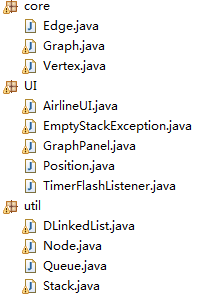
**2.1数据结构**

采用双链表的方式来表示点集、边集。

点的类中包含一个和它相连的边的集合，可以查到所有和它邻接的其他的点。

**2.2方法与类简介**

**2.2.1类的结构**

****

主类是UI中的AirlineUI,它包含了main方法，以及主要的界面元素，和Listener（监听器），GraphPanel是中间用于画图的panel，AirlineUI中包含了一个static的Graph类变量，使所有类可以看到这个变量。

Graph中封装了graph用得到的方法和变量：如点集、计算最短路径的方法、删除、添加边的方法，删除、添加点的方法，删除所有图的元素，清空图的方法，读写文件的方法等。其中提供的添加、删除点、边的方法都有自己用的传入对象的方法和重写的传入字符串的用户专用的包装过的方法，实现了内外部的分离。

Vertex中包含了邻接的边集，删除所有邻接的边集，拿到相邻的点的方法，edge处于附属在点上的地位。以避免混乱的结构也便于dijkstra方法的计算。提供了较好的结构。

由于问题的特殊性，写了Vertex和Edge的equal方法，代替原来的比较方法。便于准确的找到点、边。

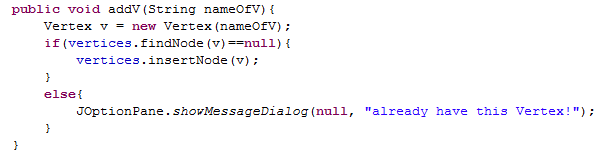
而一些数据结构类放在util包中。DLinkedList是最主要的数据结构（即双链表），它提供了拿到里面的元素个数（length），根据定义的比较方法找到元素、添加、删除节点的功能。Graph类中的添加、删除方法包装了这里定义的方法。

**2.2.2主要方法**

**2.2.2.1导入图**

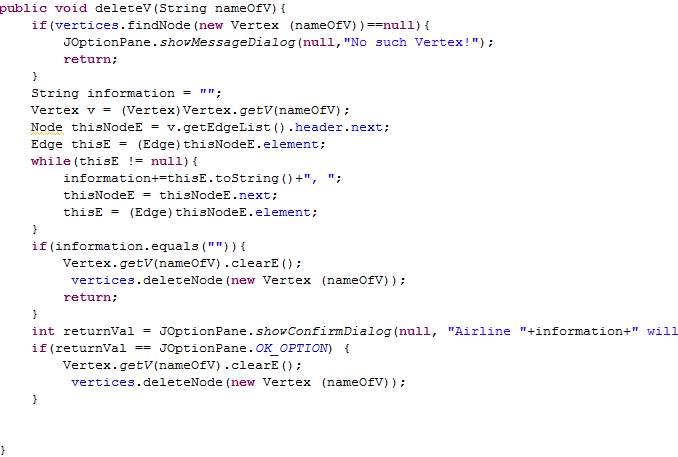
Graph中的readGraph实现此功能的算法，他读入并分析，如果是“VERTEX”则新建这个名字的点，在vertices（点集）中加入点，如果是“EDGE”，则新建这个边，在这条边的起点和终点的邻接边集中加入这条边。随后把读入的File告诉主类，便于之后改变以后的存档。并且repaint（），repaint（）会遍历所有边、点并把他们画出来。

**2.2.2.2加入机场**

****

在调用此方法前会判断传入的字符串是否为空，不为空则调用此方法，尝试在点集中找到此点，如果确定不重复，才能添加。否则会警告。

**2.2.2.3删除机场**

****

在调用此方法前会判断传入的字符串是否为空，不为空则调用此方法，尝试在点集中找到此点，如果确定有这个点，才能继续。否则会警告。接下来会找到他的邻接边，并发出警告，如果删掉这个节点会由航线没有机场可落。

征得同意后会删除所有与之相连的边和这个点本身。

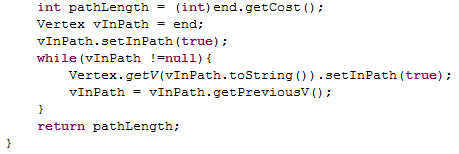
**2.2.2.4加入航线、删除航线**

原理同添加、删除点，同样会检测是否为null，然后试着找到相应的边，删除边之前会检查端点度数，如为1，则发出警告。征得同意后删除。

所有添、删动作结束后会写文件。

**2.2.2.5计算最短路径**

** **

****

这个方法基于Dijkstra算法， 每次调用先初始化，清空上次计算状态，然后从起点开始，将起点作为第一轮的frontVertex，遍历邻接边，如果从这个点走过去到达邻接点路径长可以变小，则赋值，并把那个端点的前端点设为此轮的frontVertex，找出frontVertex的所有邻接边中没有访问过的，权最小的一个，然后将那个端点作为下一轮的frontVertex，如此循环，直到所有点都被访问到，如果一个点的邻接点均已访问，则回朔，如果回朔到起点还不能将所有点遍历，则说明图不连通，会发出警告。遍历结束后，会收集整理出路径，画出来，并打印出最短路径长度。